SPECIFICATION

TO WHOM IT MAY CONCERN:

Be it known that we, with names, residence, and citizenship listed below, have invented the inventions described in the following specification entitled:

A FILTERING WEB PROXY FOR RECORDING WEB-BASED TRANSACTIONS THAT SUPPORTS SECURE HTTP STEPS

Jeffrey W. Kehoe

Residence: 3231 Fernwood Lane, Ft. Collins, CO 80525

Citizenship: United States of America

Robert Lehmann

Residence: 6430 Buffaloberry Lane, Bozeman, MT 59718

Citizenship: United States of America

A FILTERING WEB PROXY FOR RECORDING WEB-BASED TRANSACTIONS THAT SUPPORTS SECURE HTTP STEPS

Ins A1

Background of the Invention

[0001] Internet server systems are now a critical component to many successful businesses. Many Internet server systems are configured to function as e-commerce web sites where computer users can purchase goods and services. The efficient and reliable operation of the e-commerce web site is vital to many businesses.

[0002] In response to the need for efficient and reliable e-commerce web sites, test systems have been developed to ensure that the web site is operating within tolerable thresholds. These test systems perform automated tests using transactions that were previously recorded. To the web site, the test transaction appears like another customer. These test user transactions are able to determine how long a typical transaction takes and whether or not the e-commerce web site is responding at all.

[0003] Unfortunately, current test systems treat the Internet server system that provides the e-commerce web site as a black box – meaning that the test system sends in stimulus and measures response. Test systems do not effectively correlate user test results with internal performance measurements from the Internet server system. If there is a problem, the test system does not effectively isolate the responsible component within the Internet server system.

[0004] Current test systems also fail to correlate system testing and performance data with business performance data. Business performance data may only be produced in weekly or monthly reports. If the web site operator receives an alarm from

a test system, another system must be used to assess the financial damage due to the system error. The use of multiple systems is complex and time consuming.

[0005] For effective testing, the test transactions must be properly configured. As the web site changes, new features and equipment need new test transactions for testing. In a typical sequence to configure a transaction, the user operates a web browser to interact with the web site, and the web browser activity is recorded by a system in between the web browser and the web site. The recorded activity forms the test transaction that is saved for subsequent automated testing.

[0006] To put up a web site, the business often uses another entity to provide the web site infrastructure, such as an Internet Service provider (ISP), that owns and operates Internet server systems. The business must interact with the ISP to generate and implement new test transactions. Often, the business receives some client software that it operates with a web browser to generate and implement test transactions through the ISP. Different versions of the client software must be developed for the different web browsers, and possibly for the different versions of the same web browser. Unfortunately, the client software also requires the use of cookies or Java applets that can be too complex for some business users – especially since the ISP is supposed to handle the technical aspects of the web site. Cookies are files that are transferred to the web browser for local storage and use by a web server. Many people dislike storing cookies on their machines.

[0007] The cookies and Java applets are required when configuring a transaction to maintain the proper configuration sequence or state. Without proper management, a non-technical user may be easily lost in a transaction configuration sequence. The

problem becomes acute when the non-technical user begins to use forward and backward browser commands during a recording session.

[0008] Another problem during transaction recording occurs when secure Internet connections are invoked. Secure connections are often used for Internet commerce and need to be tested – especially their effects on transaction time. Configuration tools between the web browser and the web site that record web browser activity must decrypt the web browser activity to record a secure transaction. Thus, the recording component must either have access to the security keys or must be integrated with the web browser. Both of these techniques add too much complexity to the configuration tool.

Summary

[0009] The invention is a software product for testing and monitoring an Internet server system. Advantageously, the software product communicates with a web browser without the need for other client software to configure transactions. The user of the web browser is guided through web pages to record, edit, and playback transactions. Recording may occur over a secure connection. The software product performs automated tests using the transactions in addition to measuring both system performance data and business performance data. The software product generates alarms when thresholds are exceeded. The test data, performance data, and alarms are correlated in time and presented graphically to the user.

[0010] One aspect of the invention is that it allows users to record the steps of a web-based transaction (e.g. buying a book on the web) using a normal browser by

utilizing a filter proxy as a web proxy. The invention allows the filter proxy to see the HTTP request before the secure connection is made. The filter proxy is enabled to record secure requests. Other aspects of this invention include at least: 1) the filter proxy does not have to keep a list of altered secure URLs because the appended string of characters identifies the; 2) the filter proxy also provides a mechanism to differentiate new page requests from requests for embedded objects; 3) multiple proxies can be instantiated to allow multiple recording sessions; and 4) the filter proxy c an provide other filtering of the request and response (e.g. HTTP headers can be filtered).

Description of the Drawings

FIG. 1 is a block diagram that illustrates Internet server system testing and monitoring in an example of the invention.

FIG. 2 is a chart diagram that illustrates Internet server system testing and monitoring in an example of the invention.

FIG. 3 is a block diagram that illustrates test instructions in an example of the invention.

FIG. 4 is a block diagram that illustrates a web browser login page in an example of the invention.

FIG. 5 is a block diagram that illustrates a web browser select transaction page in an example of the invention.

FIG. 6 is a block diagram that illustrates a web browser record transaction page in an example of the invention.

FIG. 7 is a block diagram that illustrates a web browser edit transaction page in an example of the invention.

FIG. 8 is a block diagram that illustrates a web browser play transaction page in an example of the invention.

FIG. 9 is a state diagram that illustrates transition state rules for web browser pages in an example of the invention.

FIG. 10 is a flow diagram that illustrates recording a transaction in an example of the invention.

FIG. 11 shows a block diagram of the HTTP transaction test and the associated configuration tool.

Detailed Description of the Invention

Internet Server System Testing and Monitoring -- FIGS. 1-2

[0011] FIGS. 1-2 depict a specific example of Internet server system testing and monitoring in accord with the present invention. Those skilled in the art will appreciate numerous variations from this example that do not depart from the scope of the invention. Those skilled in the art will also appreciate that various features described below could be combined with other embodiments to form multiple variations of the invention. Those skilled in the art will appreciate that some conventional aspects of FIGS 1-2 have been simplified or omitted for clarity.

[0012] FIG. 1 is a block diagram that illustrates Internet server system testing and monitoring in an example of the invention. FIG. 1 includes Internet server system 100 and computer system 110 that have access to software product 115. Internet server

system 110 is coupled to Internet connections 105 and includes web servers 101, session servers 102, transaction servers 103, and database servers 104. Web servers 101 use web pages to interact with users over Internet connections 105. Session servers 102 track and respond to individual user activity by selecting web pages with appropriate transaction data. Transaction servers support session servers 102 by exchanging transaction data. Database servers 104 store transaction data.

[0013] Computer system 110 includes processor 111 that communicates with each of web servers 101, session servers 102, transaction servers 103, and database servers 104 over link 106. Link 106 could be implemented within Internet connections 105. Computer system 110 tests and monitors Internet server system 100 in response to processor 111 executing the instructions of software product 115.

[0014] Software product 115 comprises software storage media 120 and software storage media 130, and if desired, the two storage media could be integrated together. Software storage media 120 includes test instructions 121, system performance instructions 122, and business performance instructions 123. Software storage media 130 includes agent instructions 131. Some examples of software storage media 120 and 130 are memory devices, tape, disks, integrated circuits, and servers. Processor 111 retrieves and executes test instructions 121, system performance instructions 122, and business performance instructions 123 using software access link 112. Internet server system 100 retrieves and executes agent instructions 131 using software access link 132. Those skilled in the art appreciate that software access links 112 and 132 are logical entities that represent various structures for providing software access. For example, software product 115 could be downloaded from a server or transferred from a

disk to a local memory device in computer system 110 for subsequent retrieval and execution by processor 111. The instructions 131 are operational when executed by Internet server system 100 to direct Internet server system 100 to operate in accord with the invention. Instructions 121-123 are operational when executed by processor 111 to direct processor 111 to operate in accord with the invention. The term "processor" refers to a single processing device or a group of inter-operational processing devices. Some examples of processor 111 include computers, integrated circuits, and logic circuitry. Those skilled in the art are familiar with instructions, processors, and storage media.

[0015] Test instructions 121 direct processor 111 to configure and execute user transaction tests and report user transaction test results. User transaction tests emulate user activity to generate test results such as transaction times and data transfer rates. System performance instructions 122 direct processor 111 to measure system performance data for each of web servers 101, session servers 102, transaction servers 103, and database servers 104. Examples of system performance data include processing capacity and data retrieval time. Business performance instructions 123 direct processor 111 to measure business performance data. Examples of business performance data include: 1) monetary volume transacted by Internet server system 100 during a time period, 2) new orders transacted by Internet server system 100 during a time period, 3) sales volume for an item transacted by Internet server system 100 during a time period, 4) lost sales due to system errors during a time period, and 5) abandoned shopping carts during a time period. Agent instructions 131 direct Internet

server system 100 to read its log files to collect and transfer the system performance data and the business performance data to processor 111.

[0016] FIG. 2 is a chart diagram that illustrates Internet server system testing and monitoring in an example of the invention. FIG. 2 illustrates a chart that is generated by processor 111 and that could be included in a graphical display or report. The horizontal axis represents time of day and day of week. The vertical axis represents both dollars per hour transacted at Internet server system 100 and the performance of session servers 102 performance. Both are correlated in time along the horizontal axis. Performance could be processing load or some other system performance metric. The solid lines represent actual system performance data and actual dollars per hour performance data. The lines for the actual performance data are bounded by upper and lower thresholds that can be set by the user to trigger alarms. The lines for the actual performance data have corresponding baselines which represent averages for the time of day and day of week that have been previously calculated. From the chart, one can readily deduce that poor session server performance caused a serious loss in dollars per hour on Tuesday morning. Alarms were generated when the actual performance data fell below the lower thresholds. Alarms could trigger phone calls, e-mails and pages to significant personnel who can quickly pull up the chart using a remote terminal.

[0017] System performance instructions 122 and business performance instructions 123 direct processor 111 to correlate the system performance data and the business performance data in time. System performance instructions 122 direct processor 111 to process the system performance data to generate system performance averages associated with time of day and day of week. System

performance instructions 122 direct processor 111 to process the system performance data to generate system alarms when a system performance threshold associated with time of day and day of week is exceeded. System performance instructions 122 direct processor 111 to process the system performance data to generate system graphics illustrating system performance measured against baselines and thresholds

[0018] Business performance instructions 123 direct processor 111 to process the business performance data to generate business performance averages associated with time of day and day of week. Business performance instructions 123 direct processor 111 to process the business performance data to generate business alarms when a business performance threshold associated with time of day and day of week is exceeded. Business performance instructions 123 direct processor 111 to process the business performance data to generate business graphics illustrating business performance measured against baselines and thresholds.

Test Instructions and Computer System Operation – FIGS 3-10

[0019] FIGS. 3-10 depict a specific example of test instructions and computer system operation in accord with the present invention. Those skilled in the art will appreciate numerous variations from this example that do not depart from the scope of the invention. Those skilled in the art will also appreciate that various features described below could be combined with other embodiments to form multiple variations of the invention. Those skilled in the art will appreciate that some conventional aspects of FIGS 3-10 have been simplified or omitted for clarity.

[0020] FIG. 3 is a block diagram that illustrates test instructions 121 in an example of the invention. FIG. 3 also shows user device 140 that allows a user to operate web browser 141 to display web pages and input data. Web browser communicates with processor 111 over link 142. Typically, links 106 and 142 include a firewall to protect computer system 110.

[0021] In response to processor 111 executing test instructions 121, computer system 110 operates to configure a transaction for the user operating web browser 141. The transaction is used for automated testing of Internet server system 100. One example of a transaction is a purchase from Internet server system 100. Test instructions 121 include transaction configuration instructions 150, page transition instructions 151, proxy instructions 152, request instructions 153, and response instructions 154. Test instructions 121 direct processor 111 to: 1) interact with web browser 141 and Internet server system 100 to record web browser activity to generate the transaction, 2) edit the transaction, 3) perform an automated test of Internet server system 100 using the transaction to validate the transaction for subsequent automated testing, 4) display test results from the automated test to the user, 5) save the transaction for subsequent automated testing of Internet server system, and 6) periodically perform automated tests of Internet server system 100 using the transaction and report test results. Transaction save operations should not require system restart.

[0022] Test instructions 121 may direct processor 111 to interact with web browser 141 and Internet server system 100 through a firewall. Test instructions 121 direct processor 111 to record the web browser activity to generate test measurements, such as the sequence of web pages. Test instructions 121 direct processor 111 to add

test measurements, such as transaction time and transaction data transfer rate, to the transaction.

[0023] Test instructions 121 can also direct processor 111 to record the browser activity as a series of steps and to edit the transaction to specify test measurements for each step. Examples of test measurements for a step include elapsed time, a required string in an Internet server system response, and a prohibited string in an Internet server system response. Test instructions 141 may also direct processor 111 to record pauses for the steps and edit the transaction to redefine the pauses.

[0024] Test instructions 121 have the following test configuration and performance features. They can handle HTTP frames, cookies, and secure connections. They can parse pages for content and provide notification if content changes are discovered. They can support nested transactions. They can specify ports for use by test performance systems. They can emulate user data input, such as data fills, check boxes, and button clicks. They can handle dynamic transaction updates, and they may be written in Java.

[0025] Transaction configuration instructions 150 direct processor 111 to generate and transfer Hypertext Markup Language (HTML) pages without cookies to web browser 141. Advantageously, a standard web browser can be used to remotely configure a complex test transaction without storing cookies or Java applets on user device 140. Transaction configuration instructions 150 direct processor 111 to configure the transaction for automated testing of Internet server system 100 in response to user inputs to the HTML pages. The HTML pages include a user login

page, a transaction selection page, a transaction record page, a transaction edit page, and a transaction play page. The pages can be set-up using different languages.

[0026] FIG. 4 is a block diagram that illustrates web browser login page 460 in an example of the invention. Page 460 includes data entry boxes for user name and password that is verified by computer system 110 to authorize the user. Page 460 includes logout and help buttons. Address checks may be used to provide additional security.

[0027] FIG. 5 is a block diagram that illustrates web browser select transaction page 560 in an example of the invention. Page 560 allows the user to use web browser 141 to select a transaction. Page 560 includes new transaction data entry boxes for transaction name, first transaction step name, and a Uniform Resource Locator (URL) for the first transaction step. A record button corresponds to the new transaction data. An existing transaction drop-down selection box corresponds to buttons for edit, play, delete, rename, monitor, and un-monitor. Monitor initiates periodic testing with the transaction, and un-monitor stops periodic testing with the transaction. Page 560 also includes logout and help buttons.

[0028] FIG. 6 is a block diagram that illustrates web browser record transaction page 660 in an example of the invention. Page 660 allows the user to use web browser 141 to initiate a recording of web browser activity to generate the transaction. Page 660 identifies the transaction, the transaction step, the URL for the transaction step, and a pause for the URL. Page 660 displays the web page of the URL. Page 660 has buttons for refresh, stop, insert step, and help.

[0029] FIG. 7 is a block diagram that illustrates web browser transaction edit page 760 in an example of the invention. Transaction edit page 760 allows the user to use web browser 141 to edit the transaction generated using transaction record page 660. Transaction edit page 760 identifies the transaction and includes data entry boxes for the transaction steps, page titles for each step, URLs for each step, a pause for each URL, and test conditions for each URL. Test conditions include must have strings and fails with strings for each page. Test edit page 760 includes buttons for record, play, stop, save, insert step, delete, and help. A check box for play with pauses is included. If desired, page 760 could allow recorded user data inputs for each step to be viewed and edited.

[0030] FIG. 8 is a block diagram that illustrates web browser play transaction page 860 in an example of the invention. Transaction play page 860 allows the user to use web browser 141 to view results of an automated test using the transaction generated using transaction record page 660 and edited using transaction edit page 760. Page 860 has buttons for refresh, stop, and help. Transaction play page 860 identifies the transaction, the transaction steps, and test results for each of the transaction steps. Test results include valid page content, test complete, and error messages. Valid page content is determined by searching each page to find must have strings and determining an absence of fails with strings.

[0031] FIG. 9 is a state diagram that illustrates transition state rules 961 for web browser page transitions in an example of the invention. Page transition instructions 151 direct processor 111 to transition between the pages in response to the user inputs and to constrain the transition between the pages based on transition state rules 961.

Transition state rules 961 constrain the transition between the pages to: 1) transition from transaction selection page 560 to transaction record page 660 in response to a selection page record request, 2) transition from transaction record page 660 to transaction edit page 760 in response to a record page stop request, 3) transition from transaction edit page 760 to transaction play page 860 in response to an edit page play request, 4) transition from transaction play page 860 to transaction edit page 760 in response to a play page stop request. Transition state rules 961 may also constrain the transition between the pages 460-860 to: 1) start at user login page 460 and transition to transaction selection page 560 in response to an authorized login, 2) transition from transaction selection page 560 to transaction edit page 760 in response to a selection page edit request, 3) transition from transaction selection page 560 to transaction play page 860 in response to a selection page play request, 4) transition from transaction edit page 760 to transaction record page 660 in response to an edit page record request, 5) transition from transaction edit page 760 to transaction selection page 560 in response to an edit page stop request, 6) transition from transaction selection page 560 to user login page 460 in response to a selection page stop request, and 7) transition from transaction play page 860 to transaction selection page 560 in response to a play page stop request and the transition from transaction selection page 560 to transaction play page 860.

[0032] FIG. 10 is a flow diagram that illustrates computer system 110 operation when recording a transaction. Proxy instructions 152 direct processor 111 to receive a first request from web browser 141, transfer the first request to the Internet, and receive a response to the first request from the Internet. Response instructions 154 direct

processor 111 to search the response for a secure address, and if the response includes the secure address, then to replace the secure address with a non-secure address and identifying characters. Response instructions 154 direct processor 111 to search the first response for embedded objects, and if the response includes any embedded objects, then to add corresponding embedded object addresses to a list.

[0033] Proxy instructions 152 direct processor 111 to transfer the response to web browser 141 and receive a second request from web browser 141. Request instructions 153 direct processor 111 to record the second request as a new page if the second request is not for any of the embedded object addresses on the list. Request instructions 153 direct processor 111 to clear the list if the second request is not for any of the embedded object addresses on the list. Request instructions 153 direct processor 111 to replace the non-secure address and the identifying characters with the secure address if the second request is for the non-secure address and the identifying characters. Proxy instructions 152 direct processor 111 to transfer the second request to the Internet.

[0034] Typically, the requests comprise Hypertext Transfer Protocol requests, the secure address and the non-secure address comprise URLs, and the response comprises a Hypertext Markup Language page. Typically, request instructions 153 direct processor 111 to record: 1) URLs for new page requests, 2) the sequence of the new page requests, 3) elapsed time between new page requests, 4) user input within the new page requests. In some examples of the invention, response instructions 153 direct processor 111 to search a header in the first response for a special instruction, and if the header includes the special instructions, then to record the special instruction.

[0035] A system of the invention for testing a web transaction has the ability to test a sequence of web pages, i.e. steps, as part of a transaction. The invention also provides support for nested transactions. For example, the "buy a book" transaction could have nested transactions of "select a book", "add to shopping cart", and check out". Further, each step of a transaction can have one or more web pages associated with it. Each step of a transaction has all of the standard HTTP measurements available to it, e.g. TotalREsponseTime, DnsTime, DataTransferRate, ValidPageContent, etc., as well as the full configuration options of a standard test. HTTP forms may also be filled in (i.e. entering data, selecting check-boxes and/or menu items, clicking on links/buttons, etc.). The invention also allows for page content change notifications which may occur via a content checking mechanism in the HTTP test. The invention also handles dynamic update, thus requiring not restarting of the agent for transaction changes. Finally, the invention also supports simulation of user pauses within a transaction.

[0036] A system of the invention for configuring a transaction provides for no programming or scripting. A browser-based interface is used to record a transaction. A proxy will record the transaction. A browser-based interface is used to define/edit transactions. Transaction configuration can be done remotely through firewalls. Transactions can be played back to provide for verification. Editing a transaction does not require a restart of the agent.

[0037] FIG. 11 shows a block diagram of the HTTP transaction test and the associated configuration tool. The configuration tool 1100 includes a browser-based configuration tool 1101, a configuration servlet 1102, a configuration proxy 1103 and a

DMS 1104 (in particular, the SMM 1105 which is utilized in configuration editing. The configuration tool is a web browser-based interface made up of a control panel 1106 for configuring transactions, a step editor 1107 for editing steps within a transaction and a page content 1108 which displays the interaction with the actual web site being tested.

[0038] In a preferred embodiment of the invention, a substantial amount of the processing is performed within the configuration servlet 1102 and the configuration proxy 1103. The configuration servlet 1102 provides the actual HTML-based interaction with the user for the control panel 1106 and step editor 1107 within the configuration tool 1100. In one embodiment of the invention (not shown), the configuration servlet 1102 and the configuration proxy are combined into a single element. FIG. 11 shows the configuration servlet 1102 and the configuration proxy 1103 as separate logical components. Configuration changes are passed to the SMM via the discovery interface 1109. This allows run-time adds, deletes, and changes of transactions and their associated measurements. The transaction edits are then propagated down to the agent via the standard agent configuration path.

[0039] Considering the aspects of the invention directed to a filtering web proxy for recording web-based transactions, a user configures a web browser to use the filter proxy as its web proxy with no direct web connections. All web transactions then go through the filter proxy, which records the pages requested. The filter proxy parses the response. When a reference to a secure URL is found (e.g. a redirection, hyperlink, etc.), the filter proxy changes the reference to a non-secure reference. The filter proxy appends an identifying string of characters to the URL. By doing so, the filter proxy does not have to track which URLs have been altered. If a URL is requested that was

previously altered by the proxy filter, the filter proxy identifies it by the appended characters and changes it back to a secure request.

[0040] Further, the filter proxy has SSL libraries to establish a secure connection to the server and complete the transaction. Since the proxy is parsing the HTML while looking for secure references, it also has the ability to identify embedded objects. Identification of embedded objects allows for differentiation of requests for new pages from embedded objects. The proxy keeps a list of embedded objects. The objects can be kept in any type of file or list. When the proxy receives a request, it checks the list for the requested URL. If the URL is in the list, it is an embedded object and not a request for a new page that sho8uld be recorded. If the URL is not in the list, the request is for a new page. The URL is then recorded and the list is cleared.

[0041] As a result, the invention is not limited to the specific examples and illustrations discussed above, but only by the following claims and their equivalents.